AMENDMENTS TO THE SPECIFICATION

Please amend pages 8 and 9 of the specification as follows:

According to an embodiment of the present invention, these error checking NOP instructions can be utilized without the implementation of hardware (i.e., the NOP PSR generator). There are several ways this could be implemented: 1) A compiler could maintain heuristics as to the lifetime of various pieces of state information, and could insert specific NOPs into the program instructions to test those pieces of state. In this way, there would be enhanced error protection incorporated seamlessly and transparently in the program. 2) A post processor could scan the resulting code stream, altering the standard NOPs that were inserted by the compiler, creating a new code stream in a separate operation. Since this post-processing program would not necessarily have the same visibility into the actual execution of the program that the compiler could, it would choose NOPs that appear more random in their testing impact. 3) The operating system program loader could implement the post processing system as each program is loaded into memory. In this case, the modified NOPs could be the same from program start to program start or could vary between loads.

According to this embodiment of the present invention, these implementations complement the VLIW type of architectures because of the opportunities present in the architecture. For example, in VLIW architectures, there is a natural background of NOPs to take advantage of because of the difficulty of filling all computational slots. For other architectures, such as RISC or CISC, NOPs are generally undesirable because they waste computational resources. However, the same principals may be applied to these architectures as is described above. In these cases, there is a trade-off between the performance lost to the inserted NOPs and the additional security the error checking NOPs provide.

As seen from above, the use of error checking NOP instructions works to ensure a maximum error lifetime without inhibiting overall system performance within redundant processors. This is because the insertion of these numerous NOP instructions ensure the fact that the checking hardware checks more frequently and has more locations to check. Also, taking advantage of modern processor optimizations for handling NOPs minimizes any impact on processor performance.

Although several embodiments are specifically illustrated and described herein, it will be appreciated that modifications and variations of the present invention are covered by the above teachings and within the purview of the appended claims without departing from the spirit and intended scope of the invention. For example, there are many examples of neutral instructions that do not affect the architectural state of the processor. These include the following: ~~XORing~~ ORing or ANDing the contents of a memory location with itself, adding 0 to such a value, ANDing the value with 1, ORing the value with 0, etc. Whether an instruction is neutral will depend on the definition of the instruction, its operands, and how it affects memory contents including flags.